Fenchel Decomposition for Stochastic Mixed-Integer Programming

Lewis Ntaimo
Department of Industrial and Systems Engineering, Texas A&M University, 3131 TAMU, College Station, TX 77843, USA, ntaimo@tamu.edu

Abstract

This paper introduces a new cutting plane method for two-stage stochastic mixed-integer programming (SMIP) called Fenchel decomposition (FD). FD uses a class of valid inequalities termed, FD cuts, which are derived based on Fenchel cutting planes from integer programming. First, we derive FD cuts based on both the first and second-stage variables, and devise an FD algorithm for SMIP and establish finite convergence for binary first-stage. Second, we derive FD cuts based on the second-stage variables only and use an idea from disjunctive programming to lift the cuts to the higher dimension space including the first-stage variables. We then devise an alternative algorithm (FD-L algorithm) based on the lifted FD cuts. Finally, we report on computational results based on several test instances from the literature. The results are promising and show that both algorithms can outperform a standard direct solver and a disjunctive decomposition algorithm on some large-scale instances. Furthermore, the FD-L algorithm provides better performance than the FD algorithm in general.

Keywords: Stochastic programming, integer programming, Fenchel decomposition, FD cuts, lifting

1 Introduction

In this paper, we consider two-stage stochastic mixed-integer programs (SMIPs) of the following form:

$$\begin{aligned} \text{SP2: Min } & c^\top x + \mathbb{E}_{\tilde{\omega}}[f(\tilde{\omega}, x)] \\ \text{s.t. } & Ax \geq b \\ & x \in \mathbb{B}^{n_1}, \end{aligned} \tag{1}$$

where for an outcome ω of $\tilde{\omega}$, the recourse function $f(\omega, x)$ is given by

$$\begin{aligned} f(\omega, x) = \text{Min } & q(\omega)^\top y(\omega) \\ \text{s.t. } & W(\omega)y(\omega) \geq h(\omega) - T(\omega)x \\ & y(\omega) \geq 0, y_j(\omega) \in \mathbb{B}, \ \forall j \in J. \end{aligned} \tag{2}$$

In formulation (1) , x denotes the first-stage decision vector, $c \in \Re^{n_1}$ is the first-stage cost vector, $b \in \Re^{m_1}$ is the first-stage righthand side, and $A \in \Re^{m_1 \times n_1}$ is the first-stage constraint matrix. In the second-stage formulation (2), $y(\omega) \in \Re^{n_2}$ denotes the recourse decision vector, $q(\omega) \in \Re^{n_2}$ is the cost vector, $h(\omega) \in \Re^{m_2}$ is the righthand side, $T(\omega) \in \Re^{m_2 \times n_1}$ is the technology matrix, and $W(\omega) \in \Re^{m_2 \times n_2}$ is the recourse matrix for scenario ω. The symbol $\mathbb{B}$ denotes the set of binary integers. Subproblem (2) is generally referred to as the *scenario* problem.

Formulation (1-2) can also be written in extensive form (deterministic equivalent) as follows:

$$\begin{aligned}
\text{EF2: } & \text{Min } c^\top x + \sum_{\omega \in \Omega} p_\omega q(\omega)^\top y(\omega) \\
& \text{s.t. } Ax \geq b \\
& \quad T(\omega)x \ + W(\omega)y(\omega) \geq h(\omega) \ \ \forall \omega \in \Omega \\
& \quad x \in \mathbb{B}^{n_1}, \ y(\omega) \geq 0, y_j(\omega) \in \mathbb{B}, \ \forall j \in J, \ \forall \omega \in \Omega.
\end{aligned} \tag{3}$$

We will denote by $\text{conv}(X)$ the convex hull of first-stage feasible solutions and by $\text{vert}(\text{conv}(X))$ the set of vertices (extreme points) of $\text{conv}(X)$. We address instances of SP2 under the following assumptions:

(**A1**) The random variable $\tilde{\omega}$ follows a discrete distribution with finite support Ω. The probability of an outcome ω of $\tilde{\omega}$ is equal to p_ω.

(**A2**) The first-stage feasible set $X = \{Ax \geq b, x \in \mathbb{B}^{n_1}\}$ is nonempty and $x \in \text{vert}(\text{conv}(X))$.

(**A3**) The second-stage feasible set $Y(\omega, x) = \{W(\omega)y(\omega) \geq h(\omega) - T(\omega)x\}$ is nonempty and bounded for all $x \in X$ and $\omega \in \Omega$.

Assumption (A1) is required to make the problem tractable. As detailed in Section 3.1, assumption (A2) is required to guarantee the existence of an optimal solution and finite convergence of the proposed cutting plane method. Finally, assumption (A3) is needed for ease of exposition of the cutting plane algorithms described in Section 3 and Section 4.

Even though MIP solvers can be applied directly to EF2, the large-scale nature and the integrality requirements on the decision variables make the problem very difficult to solve in general. Nevertheless, SP2 is amenable to decomposition methods that exploit the special structure inherent in the problem formulation. In the special case when the second-stage has only continuous variables, for a given $\omega \in \Omega$ the recourse function $f(\omega, x)$ is a well-behaved piecewise linear and convex function of x. Thus Benders' decomposition (Benders, 1962) is applicable in this case (Wollmer, 1980). Otherwise, when the second-stage variables involve integrality restrictions, $f(\omega, x)$ is lower semicontinuous with respect to x (Blair and Jeroslow, 1982), and is generally nonconvex (Schultz, 1993). The focus of this paper is on SP2 with $x \in \text{vert}(\text{conv}(X))$ such as binary first-stage, and mixed-integer second-stage. In SMIP, the type of decision variables (continuous, binary, integer, mixed-integer) and in which stage they appear, greatly influences algorithm design.

We introduce a new cutting plane method for SP2 called *Fenchel Decomposition* (FD). FD is based on a new class of cutting planes termed *FD cuts*, which we derive based on Fenchel cuts from integer programming (Boyd, 1993, 1994a,b, 1995). We first derive FD cuts based on the $(x, y(\omega))$-space of the EF2 structure and devise an FD algorithm for SP2. We then derive an alternative class of FD cuts based on the $y(\omega)$-space. We use an idea from disjunctive programming (Balas et al., 1993) to lift the FD cuts to the $(x, y(\omega))$-space. Fenchel cutting planes are based on the ability to optimize a linear function over a polyhedron rather than explicit knowledge of the facial structure of the polyhedron. The name 'Fenchel cut' comes from the similarities with Fenchel duality regarding the maximum separation/minimum distance duality in convexity theory (Rockafellar, 1970). Even though the stochastic counterpart of these cuts have not been investigated before, Fenchel cuts were demonstrated to be effective in solving several classes of integer programs (Boyd, 1993, 1994a,b). Sáez (2000) investigated solving linear programming relaxations associated with Lagrangian relaxations by Fenchel cutting planes with great success. More recently, Ramos and Sáez (2005) implemented Fenchel cuts to solve the deterministic capacitated facility location problem. Several characteristics of Fenchel cuts have been proved such as providing 'deepest' cuts or being facet defining (Boyd, 1994a), and finite convergence of the cutting plane method (Boyd, 1995). However, Fenchel cuts can be computationally expensive in general and are best suited for problems with special structure. For example, Boyd (1993, 1995) exploited the structure of knapsack problems with nonnegative left-hand side coefficients.

Algorithms for SMIP is still an active area of research probably due to the computational challenges posed by SP2 and its many real life applications. Several surveys on algorithms for SMIP (Schultz et al., 1996, Klein Haneveld and van der Vlerk, 1999, Louveaux and Schultz, 2003, Schultz, 2003, Sen, 2005) and a few textbooks are available (Birge and Louveaux, 1997, Ruszczyn'ski and Shapiro, 2003, Shapiro et al., 2009). Closely related cutting plane methods for SMIP include the method by Carøe and Tind (1997), Carøe (1998) and the disjunctive decomposition ($D2$) method (Sen et al., 2002, Sen and Higle, 2005). Both methods are based on disjunctive programming (Balas, 1975, Blair and Jeroslow, 1978, Sherali and Shetty, 1980) and provide a rather general setting for the study of the convex hull of feasible points of SMIP problems under the two-stage setting. Extensive computations based on disjunctive cuts for SMIP are reported in (Ntaimo and Tanner, 2008).

Branch-and-cut methods include the $D2$-BAC algorithm (Sen and Sherali, 2006) and the DBAB method (Sherali and Zhu, 2006). The $D2$-BAC requires pure binary first-stage while the DBAB method allows for mixed-integer variables in both stages. The DBAB method uses Benders' (Benders, 1962) subproblems to define second-stage lower bounding value functions of the first-stage variables. Preliminary computational results with the $D2$-BAC algorithm are reported in Ntaimo and Sen (2008). A finite branch-and-bound algorithm for devised by Ahmed et al. (2004) considers two-stage stochastic integer programs with general integer first-stage. We refer the reader to a comprehensive bibliography on SMIP online by van der Vlerk (2007). Here we simply highlight some closely related cutting plane methods for SMIP.

The rest of the paper is organized as follows. We begin with some preliminaries on FD and derive FD cuts based on the structure of EF2 in Section 2. We derive a basic FD algorithm and an FD cut generation subroutine in Section 3. In Section 4 we derive lifted FD cuts and devise an alternative algorithm and cut generation subroutine. We present computational results based on several test instances from the literature in Section 5. We end the paper with concluding remarks and directions for future research in Section 6.

2 Preliminaries

We begin with the basic theory for Fenchel decomposition and derive FD cuts for SP2 based on the structure of the EF2. Consider SP2 in extensive form with the integer restrictions on the second-stage decision variables relaxed as follows:

$$\begin{aligned} \text{Min } & c^\top x + \sum_{\omega \in \Omega} p_\omega q(\omega)^\top y(\omega) \\ \text{s.t. } & Ax \geq b \\ & T(\omega)x \ + W(\omega)y(\omega) \geq h(\omega) \ \ \forall \omega \in \Omega \\ & x \in \mathbb{B}^{n_1}, \ y(\omega) \geq 0, \ \forall \omega \in \Omega. \end{aligned} \tag{4}$$

Let the feasible set of the subproblem LP relaxation for a given scenario $\omega \in \Omega$ be defined as

$$Y_{LP}(\omega) = \ \big\{x, y(\omega) \in \Re_+^{n_2} \mid Ax \geq b, \ T(\omega)x + W(\omega)y(\omega) \geq h(\omega), \ x \in \mathbb{B}^{n_1}\big\}.$$

Then the set of subproblem feasible solutions for ω can be given as

$$Y_{IP}(\omega) = \ \big\{x, y(\omega) \in Y_{LP}(\omega) \mid x \in \mathbb{B}^{n_1}, \ y_j(\omega) \in \mathbb{B}, \ \forall j \in J\big\}.$$

Also, let $Y_{IP}^c(\omega)$ denote the convex hull of $Y_{IP}(\omega)$ or conv$(Y_{IP}(\omega))$.

THEOREM 2.1. *Let $(\hat{x}, \hat{y}(\omega)) \in Y_{LP}(\omega)$ be given. Define $g(\omega, \alpha(\omega), \beta(\omega)) = Max \, \{\alpha(\omega)^\top x + \beta(\omega)^\top y(\omega) \mid (x, y(\omega)) \in Y_{IP}^c(\omega)\}$ and let $\delta(\omega, \alpha(\omega), \beta(\omega)) = \alpha(\omega)^\top \hat{x} + \beta(\omega)^\top \hat{y}(\omega) - g(\omega, \alpha(\omega), \beta(\omega))$. Then there exists vectors $\alpha(\omega)$ and $\beta(\omega)$ for which $\delta(\omega, \alpha(\omega), \beta(\omega)) > 0$ if and only if $(\hat{x}, \hat{y}(\omega)) \neq Y_{IP}^c(\omega)$.*

Theorem 2.1 follows from a basic theorem for generating a Fenchel cut in integer programming (Boyd, 1994a) and so we omit the proof. Here we apply the theorem to SP2 to allow for generating valid inequalities for $Y_{IP}^c(\omega)$ for a given scenario $\omega \in \Omega$. The inequality $\alpha(\omega)^\top x + \beta(\omega)^\top y(\omega) \leq g(\omega, \alpha(\omega), \beta(\omega))$ is valid for $Y_{IP}^c(\omega)$ and separates $Y_{IP}^c(x, \omega)$ from $(\hat{x}, \hat{y}(\omega))$ if and only if $\delta(\omega, \alpha(\omega), \beta(\omega)) > 0$. We refer to such an inequality as a *Fenchel decomposition cut* or *FD cut.*

Observe that the function $\delta(\omega, \alpha(\omega), \beta(\omega))$ is piecewise linear and concave since the value $\alpha(\omega)^\top \hat{x} + \beta(\omega)^\top \hat{y}(\omega)$ is a linear function of $\alpha(\omega)$ and $\beta(\omega)$ and the function

$g(\omega, \alpha(\omega), \beta(\omega))$ is piecewise linear and convex. This implies that generalized programming or subgradient optimization can be used to maximize $\delta(\omega, \alpha(\omega), \beta(\omega))$ if $\alpha(\omega)$ and $\beta(\omega)$ are chosen from some convex set $\Pi^{\alpha,\beta}$. Recall that for a concave function $f : \mathbb{R}^m \mapsto \mathbb{R}$ a subgradient $s \in \mathbb{R}^m$ of f at $\bar{x} \in \mathbb{R}^m$ must satisfy $f(x) - f(\bar{x}) \leq s(x - \bar{x})$. Furthermore, we can compute the subgradient of $g(\omega, \alpha(\omega), \beta(\omega))$ using the following proposition.

PROPOSITION 2.2. *Let $(\hat{x}, \hat{y}(\omega)) \in Y_{LP}(\omega)$ be given and let $(\bar{x}, \bar{y}(\omega)) \in Y^c_{IP}(\omega)$ satisfy $g(\bar{\alpha}(\omega), \bar{\beta}(\omega), \omega) = \bar{\alpha}(\omega)^\top \bar{x} + \bar{\beta}(\omega)^\top \bar{y}(\omega)$. Then $[(\bar{x} - \hat{x})^\top; (\bar{y}(\omega) - \hat{y}(\omega))^\top]^\top$ is a subgradient of $g(\omega, \alpha(\omega), \beta(\omega))$ at $(\bar{\alpha}(\omega), \bar{\beta}(\omega))$.*

In the cutting plane approach we propose, we are interested in sequentially generating (or at least partially) $Y^c_{IP}(\omega)$ for each $\omega \in \Omega$. But $Y^c_{IP}(\omega)$ appears in the definition of $g(\omega, \alpha(\omega), \beta(\omega))$ in Theorem 2.1. Therefore, we will instead work with $\mathrm{conv}\{y(\omega) \mid y(\omega) \in Y_{IP}(\omega)\}$. For a given non-integer point $(\hat{x}, \hat{y}(\omega)) \in Y_{LP}(\omega)$ the following problem can be used to generate a valid inequality for $Y_{IP}(\omega)$:

$$\delta(\omega) = \underset{(\alpha(\omega),\beta(\omega)) \in \Pi^{\alpha,\beta}}{\mathrm{Max}} \left\{ \alpha(\omega)^\top \hat{x} + \beta(\omega)^\top \hat{y}(\omega) - g(\omega, \alpha(\omega), \beta(\omega)) \right\} \tag{5}$$

where,

$$g(\omega, \alpha(\omega), \beta(\omega)) = \mathrm{Max} \left\{ \alpha(\omega)^\top x + \beta(\omega)^\top y \mid \ (x, y(\omega)) \in Y^c_{IP}(\omega) \right\}.$$

The FD cut $\alpha(\omega)^\top x + \beta(\omega)^\top y(\omega) \leq g(\omega, \alpha(\omega), \beta(\omega))$ cuts off the non-integer point $(\hat{x}, \hat{y}(\omega)) \in Y_{LP}(\omega)$ if $\delta(\omega) > 0$. In this case we can append the cut to $Y_{LP}(\omega)$. Notice that the FD cut cuts off the largest distance between the point $(\hat{x}, \hat{y}(\omega))$ and the cut based on $\Pi^{\alpha,\beta}$. Candidate domains for $\Pi^{\alpha,\beta}$ are L^1, L^2 and L^∞ norms. However, since problem (5) has to be solved many times to generate FD cuts, a linearly constrained domain for $\Pi^{\alpha,\beta}$ provides a better choice in terms of solution time.

3 Fenchel Decomposition

We are now in a position to devise an FD cutting plane method for SP2. We adopt a Benders decomposition (Benders, 1962) framework so that LP relaxation (4) is solved using the L-shaped method (Slyke and Wets, 1969). Note that the master program is kept as an integer program but one can envision an algorithm where the master program is also relaxed. Our interest is in studying the effectiveness of the FD cuts focusing on the second-stage.

3.1 FD Algorithm

Let t and k denote algorithm iteration indices. Let $\mathcal{C}_k(\omega)$ denote the set of iteration indices up to k where an FD cut is generated and added to the subproblem for scenario

ω. Then the master program at iteration k takes the form:

$$\begin{aligned} \text{Min} \ & c^\top x + \eta \\ \text{s.t.} \ & Ax \geq b \\ & \sigma_t^\top x + \eta \leq \nu_t, \ t = 1, \cdots, k \\ & x \in \mathbb{B}^{n_1}. \end{aligned} \tag{6a}$$

Constraints (6a) are the usual *optimality* cuts. For each outcome $\omega \in \Omega$, the scenario subproblem relaxation takes the following form:

$$\begin{aligned} f_c^k(\omega, x) = \text{Min} \ & q(\omega)^\top y \\ \text{s.t.} \ & W(\omega) y \ \geq r(\omega) - T(\omega) x \\ & \beta^t(\omega)^\top y \geq g(\alpha^t(\omega), \beta^t(\omega), \omega) - \alpha^t(\omega)^\top x, \ t \in \mathcal{C}_k(\omega) \\ & y \geq 0. \end{aligned} \tag{7}$$

The second set of constraints in (7) are the FD cuts generated and added to the scenario problem by iteration k. A basic FD algorithm for SP2 can then be given as follows:

Basic FD Algorithm:

Step 0. Initialization.
Set $k \leftarrow 1$, $L_1 \leftarrow -\infty$, $U_1 \leftarrow \infty$, and let $\epsilon > 0$ be given.

Step 1. Solve LP Relaxation. Solve LP relaxation (4) using the L-shaped method to get solution $(x^k, \{y^k(\omega)\}_{\omega\in\Omega})$ and objective value z^k. Set $L^k = z_k$. If $(x^k, \{y^k(\omega)\}_{\omega\in\Omega})$ satisfy the integer restrictions, set x^*=x^k, $U^k = z^k$ and stop, x^* is optimal. Otherwise, set $k \leftarrow k + 1$ and go to Step 3.

Step 2. Solve Subproblem LPs.
Solve subproblem (7) for all $\omega \in \Omega$. If $(x^k, \{y^k(\omega)\}_{\omega\in\Omega})$ satisfy integer restrictions, set $U^{k+1} \leftarrow \text{Min}\{c^\top x^k + \mathbb{E}[f_c^k(x^k, \tilde{\omega})], \ U^k\}$ and if U^{k+1} is updated, set incumbent solution $x_\epsilon^* \leftarrow x^k$, and go to Step 5.

Step 3. Solve FD cut Generation Subproblems and Add Cuts. For all $\omega \in \Omega$ such that $(x^k, y^k(\omega))$ is non-integer, form and solve (5) to obtain $\alpha^k(\omega)$, $\beta^k(\omega)$ and $g(\alpha^k(\omega), \beta^k(\omega), \omega)$. Form FD cut $\alpha^k(\omega)^\top x + \beta^k(\omega)^\top y \leq g(\alpha^k(\omega), \beta^k(\omega), \omega)$ and append to subproblem (7) for ω.

Step 4. Re-Solve Subproblem LPs.
Re-solve subproblem (7) for all $\omega \in \Omega$ for which an FD cut was added. If $(x^k, \{y^k(\omega)\}_{\omega\in\Omega})$ satisfy integer restrictions, set $U^{k+1} \leftarrow \text{Min}\{c^\top x^k + \mathbb{E}[f_c^k(\omega, x^k)], \ U^k\}$. If U^{k+1} is updated, set incumbent solution $x_\epsilon^* \leftarrow x^k$. Go to Step 5.

Step 5. Update and Solve the Master Problem.
Compute an *optimality cut* using the dual multipliers from the most recently solved subproblem and add to the master program (16). Solve the master program to

get optimal solution x^{k+1} and optimal value z^{k+1}. Set $L^{k+1} \leftarrow \text{Max}\{z^{k+1},\ L^k\}$. If $U^{k+1} - L^{k+1} \le \epsilon$, stop and declare x^*_ϵ ϵ-optimal. Otherwise, $k \leftarrow k+1$ and repeat from Step 2.

Clearly the convergence of the Basic FD algorithm to an optimal solution depends on the ability to generate facets of the scenario subproblems. We give a formal statement regarding the finite termination of the Basic FD algorithm after we state the FD cut generation subroutine in the next subsection.

3.2 FD Cut Generation

To generate an FD cut in Step 3 of FD Algorithm, we need to solve problem (5). As pointed out earlier, problem (5) can be solved using generalized programming or subgradient optimization. We consider solving problem (5) using a cutting plane decomposition approach. We first make the observation that based on Proposition 2.2, $\delta^k(\omega)$ can be maximized by solving the following problem:

$$\begin{aligned} \delta^k(\omega) = \underset{\alpha(\omega),\beta(\omega)\in\Pi^{\alpha,\beta}}{\text{Max}} \quad & \theta \\ \text{s.t.} \quad & -\theta + (x^k - x^t)^\top \alpha(\omega) + (y^k(\omega) - y^t(\omega))^\top \beta(\omega) \ge 0, \\ & (x^t, y^t(\omega)) \in \text{vert}(Y^c_{IP}(\omega)). \end{aligned} \tag{8}$$

We introduce the free variable θ in (8) to enable us to compute the maximum distance based on $\Pi^{\alpha,\beta}$ between the non-integer point $(x^k, y^k(\omega))$ and the FD cut. Since we are after recovering (at least partially) $Y^c_{IP}(\omega)$, we use $\text{conv}\{y(\omega) \mid y(\omega) \in Y_{IP}(\omega)\}$ and solve problem (8) via a decomposition cutting plane method with a master and subproblem. We consider maximizing $\delta^k(\omega)$ over a linearly defined domain $\Pi^{\alpha,\beta}$.

Let t and τ denote iteration indices for the cut generation subroutine. Then given $(x^k, y^k(\omega)) \in Y_{LP}(\omega)$, at iteration τ the master program takes the following form:

$$\delta^k_\tau(\omega) = \underset{\alpha(\omega),\beta(\omega)\in\Pi^{\alpha,\beta}}{\text{Max}} \quad \theta \tag{9a}$$

$$\text{s.t.} \quad -\theta + (x^k - x^t(\omega))^\top \alpha(\omega) + (y^k(\omega) - y^t(\omega))^\top \beta(\omega) \ge 0,\ t = 1, \cdots, \tau. \tag{9b}$$

Given an optimal solution $(\theta^t, \alpha^t(\omega), \beta^t(\omega))$ to (9) at iteration t, $(x^t(\omega), y^t(\omega))$ is the optimal solution to the following subproblem:

$$\begin{aligned} g(\alpha^t(\omega), \beta^t(\omega), \omega) = \text{Max} \quad & \alpha^t(\omega)^\top x + \beta^t(\omega)^\top y(\omega) \\ \text{s.t.} \quad & (x, y(\omega)) \in Y_{IP}(\omega). \end{aligned} \tag{10}$$

An FD cut generation subroutine can be given as follows:

FD Cut Generation Subroutine:

Step 0. Initialization.
Set $t \leftarrow 1$, $\ell^1 \leftarrow -\infty$, $u^1 \leftarrow \infty$, and choose $\epsilon' > 0$ and $(\alpha^1(\omega), \beta^1(\omega)) \in \Pi^{\alpha,\beta}$.

Step 1. Solve Subproblem and Compute Lower Bound. Use $(\alpha^t(\omega), \beta^t(\omega))$ to form and solve subproblem (10) to get solution $(x^t, y^t(\omega))$ and objective value $g_t(\alpha^t(\omega), \beta^t(\omega), \omega)$. Let $d_t = (x^k - x^t)^\top \alpha^t(\omega) + (y^k(\omega) - y^t(\omega))^\top \beta^t(\omega) - g_t(\alpha^t(\omega), \beta^t(\omega), \omega)$. Set $\ell^{t+1} \leftarrow \text{Max}\{d_t,\ \ell^t\}$. If ℓ^{t+1} is updated, set incumbent solution $(\alpha^*(\omega), \beta^*(\omega), g^*(\alpha^*(\omega), \beta^*(\omega), \omega)) = (\alpha^t(\omega), \beta^t(\omega), g_t(\alpha^t(\omega), \beta^t(\omega), \omega))$.

Step 2. Solve Master Problem. Using current non-integer solution $(x^k, y^k(\omega))$ and subproblem (10) solution $(x^t, y^t(\omega))$ to form and add constraint (9b) to master program. Solve master program to get an optimal solution $(\theta^t, \alpha^t(\omega), \beta^t(\omega))$. Set $u_{k+1} \leftarrow \text{Min}\{\theta^t,\ u^t\}$. If $u^{t+1} - \ell^{t+1} \leq \epsilon'$, stop and declare incumbent solution ϵ'-optimal. Otherwise, set $t \leftarrow t+1$ and go to Step 1.

LEMMA 3.1. *Suppose that assumptions (A1)-(A3) hold and that FD Cut Generation Subroutine uses domain $\Pi^{\alpha,\beta}$ defined by the unit sphere of an arbitrary norm. Then for a given $(x^k, y^k(\omega)) \in Y_{LP}(\omega)$ for $\omega \in \Omega$, the FD Cut Generation Subroutine terminates after a finite number of iterations.*

Lemma 3.1 follows directly from Theorem 3.2 in Boyd (1995), which shows that $\Pi^{\alpha,\beta}$ defined by the unit sphere of an arbitrary norm cannot define a face of a polyhedron more than a finite number of times. Since subproblem (10) is an integer program, generating an FD cut can be expensive. Therefore, we recommend exploiting the structure of the subproblem for a given application so that generating an FD cut is not computationally prohibitive. For example, Boyd (1995) exploits the structure of deterministic knapsack problems with nonnegative left-hand side coefficients to devise a fast Fenchel cut generation subroutine for that class of problems.

THEOREM 3.2. *Suppose that assumptions (A1)-(A3) hold and that the Basic FD algorithm uses domain $\Pi^{\alpha,\beta}$ defined by the unit sphere of an arbitrary norm and that the FD cut generated in Step 3 of the algorithm are facets of $Y_{IP}^c(\omega)$. Then the Basic FD algorithm applied to SP2 ensures that there exists a $K < \infty$ such that for all $k > K$, $f_c^k(\omega, x^k) = f(\omega, x^k)$ for all $\omega \in \Omega$.*

Proof. By Lemma 3.1, for every $(x^k, y^k(\omega)) \in Y_{LP}(\omega)$ computed in Step 2 and 5 of the Basic FD algorithm, the FD Cut Generation Subroutine in Step 3 terminates after a finite number of iterations. Since the FD cuts generated and added to $Y_{LP}(\omega)$ are facets of $Y_{IP}^c(\omega)$ and $x^k \in \text{vert}(\text{conv}(X))$, as with binary x^k, after finitely many iterations $K < \infty$ no new facets can be generated in Step 3 of the FD algorithm. Also, the x^k are finite. Therefore, at some iteration $k > K$ we must have that $f_c^k(\omega, x^k) = f(\omega, x^k)$, for all $\omega \in \Omega$. □

In the FD algorithm, the master program uses a lower bounding function to approximate $\mathbb{E}[f(\tilde{\omega}, x)]$. Thus obtaining correct subproblem objective function values in Step 4 in finitely many iterations leads to the finite termination of the FD algorithm. In theory, finite termination of the FD algorithm to an optimal solution can be guaranteed if $\Pi^{\alpha,\beta}$ is chosen to be an $n_1 + n_2$-dimensional set containing the origin in its strict interior so that $\delta(\omega)$ attains a positive value in $\Re^{n_1+n_2}$ if and only if it achieves a positive value on $\Pi^{\alpha,\beta}$ (Boyd, 1995). However, even though finite convergence of the method to an optimal solution is guaranteed under general conditions (the domain $\Pi^{\alpha,\beta}$ must be a unit sphere of an arbitrary norm), the rate of convergence can vary based on the type of norm used.

4 Lifting in Fenchel Decomposition

It may be desirable to generate FD cuts in the space of $y(\omega)$ and then lifting the cuts to the higher space $(x, y(\omega))$. This may improve computational time and/or provide better bounds within the first few algorithm iterations. We continue to assume that the master program is solved as an integer program, so the LP relaxation of SP2 takes the following form:

$$\begin{aligned} \text{Min } & c^\top x + \sum_{\omega \in \Omega} p_\omega f_c(\omega, x) \\ \text{s.t. } & Ax \geq b \\ & x \in \mathbb{B}^{n_1}, \end{aligned} \tag{11}$$

where for an outcome ω,

$$\begin{aligned} f_c(\omega, x) = \text{Min } & q(\omega)^\top y \\ \text{s.t. } & W(\omega)y \geq h(\omega) - T(\omega)x \\ & y \geq 0. \end{aligned} \tag{12}$$

Let the first-stage feasible set continue to be defined as

$$X = \{x \mid Ax \geq b, x \in \mathbb{B}^{n_1}\}.$$

Now let the subproblem LP relaxation feasible set for a given $(\omega, x) \in X \times \Omega$, be defined as

$$Y_{LP}(\omega, x) = \left\{ y(\omega) \in \Re^{n_2}_{+} \mid W(\omega)y(\omega) \geq h(\omega) - T(\omega)x \right\}.$$

Then the set of subproblem feasible solutions for (ω, x) can be given as

$$Y_{IP}(\omega, x) = \left\{ y(\omega) \in Y_{LP}(\omega, x) \mid y_j(\omega) \in \mathbb{B}, \ \forall j \in J \right\}.$$

Let $Y^c_{IP}(\omega, x) = \text{conv}(Y_{IP}(\omega, x))$ denote the convex hull of $Y_{IP}(\omega, x)$.

THEOREM 4.1. *Let for $(\omega, \hat{x}) \in \Omega \times X$, $\hat{y}(\omega) \in Y_{LP}(\omega, \hat{x})$ be given. Define $g(\omega, \hat{x}, \beta(\omega)) =$ Max $\{\beta(\omega)^\top y(\omega) \mid y(\omega) \in Y^c_{IP}(\omega, \hat{x})\}$ and let $\delta(\omega, \hat{x}, \beta(\omega)) = \beta(\omega)^\top \hat{y}(\omega) - g(\omega, \hat{x}, \beta(\omega))$. Then there exists a vector $\beta(\omega)$ for which $\delta(\omega, \hat{x}, \beta(\omega)) > 0$ if and only if $\hat{y}(\omega) \neq Y^c_{IP}(\omega, \hat{x})$.*

Theorem 4.1 is analogous to Theorem 2.1. Now we apply the theorem to enable us to generate valid inequalities for $Y_{IP}^c(\omega, x)$ for all $x \in X$ for each $\omega \in \Omega$. The inequality $\beta(\omega)^\top y(\omega) \leq g(\omega, \hat{x}, \beta(\omega))$ is valid for $Y_{IP}^c(\omega, \hat{x})$ and separates $Y_{IP}^c(\omega, \hat{x})$ from $\hat{y}(\omega)$ if and only if $\delta(\omega, \hat{x}, \beta(\omega)) > 0$. The function $\delta(\omega, \hat{x}, \beta(\omega))$ is piecewise linear and concave and generalized programming or subgradient optimization can be used to maximize $\delta(\omega, \hat{x}, \beta(\omega))$ if $\beta(\omega)$ is chosen from some convex set Π^β. We also obtain the following result that is analogous to Proposition 2.2:

PROPOSITION 4.2. *Let $\hat{y}(\omega) \in Y_{LP}(\omega, \hat{x})$ be given and let $\bar{y}(\omega) \in Y_{IP}^c(\omega, \hat{x})$ satisfy $g(\omega, \hat{x}, \bar{\beta}(\omega)) = \bar{\beta}(\omega)^\top \bar{y}(\omega)$. Then $(\hat{y}(\omega) - \bar{y}(\omega))$ is a subgradient of $g(\omega, \hat{x}, \beta(\omega))$ at $\bar{\beta}(\omega)$.*

By applying Theorem 4.1 the following problem can be used to cut off a non-integer solution $\hat{y}(\omega) \in Y_{LP}(\omega, \hat{x})$ for a given $(\omega, \hat{x}) \in \Omega \times X$:

$$\delta(\omega, x) = \underset{\beta(\omega) \in \Pi^\beta}{\text{Max}} \left\{ \beta(\omega)^\top \hat{y}(\omega) - g(\omega, \hat{x}, \beta(\omega)) \right\} \tag{13}$$

where,

$$g(\omega, \hat{x}, \beta(\omega)) = \text{Max} \left\{ \beta(\omega)^\top y(\omega) \mid y(\omega) \in Y_{IP}^c(\omega, \hat{x}) \right\}.$$

The inequality $\beta(\omega)^\top y(\omega) \leq g(\omega, \hat{x}, \beta(\omega))$ cuts off a non-integer point $\hat{y}(\omega) \in Y_{LP}(\omega)$ if $\delta(\omega, \hat{x}) > 0$.

4.1 FD Cut Lifting

Now that we have devised a method for generating an FD cut of the form $\beta(\omega)^\top y(\omega) \leq g(\omega, \hat{x}, \beta(\omega))$ for a given ω and $x \in X$, we need a method for lifting the cut to the $(x, y(\omega))$-space. This is necessary since $\beta(\omega)^\top y(\omega) \leq g(\omega, \hat{x}, \beta(\omega))$ is only valid for $Y_{IP}^c(\omega, \hat{x})$ but not for $Y_{IP}^c(\omega)$. Observe that for a given scenario $\omega \in \Omega$, the FD algorithm requires the FD cut to be valid for all $x \in X$. So we need to lift the cut $\beta(\omega)^\top y(\omega) \leq g(\omega, \hat{x}, \beta(\omega))$ to an FD cut of the form $\alpha(\omega)^\top x + \beta(\omega)^\top y(\omega) \leq g(\omega, \hat{x}, \beta(\omega)) + \alpha(\omega)^\top \hat{x}$, which is valid for $Y_{IP}^c(\omega)$. To accomplish this, we borrow the idea of facet extension or lifting from lift-and-project cuts (Balas et al., 1993), which is also used in lifting disjunctive cuts for SMIP (Carøe, 1998). We first restate the following well-known result in SMIP.

PROPOSITION 4.3. *Suppose $\hat{x} \in vert(conv(X))$, for example, $\hat{x} \in \mathbb{B}^{n_1}$. Then $conv\{y(\omega) \mid (\hat{x}, y(\omega)) \in Y_{IP}(\omega)\} = \{y(\omega) \mid (\hat{x}, y(\omega)) \in Y_{IP}^c(\omega)\}$.*

Proposition 4.3 relates $\{y(\omega) \mid (\hat{x}, y(\omega)) \in Y_{IP}^c(\omega)\}$, which appears in Theorem 4.1, to $\text{conv}\{y(\omega) \mid (\hat{x}, y(\omega)) \in Y_{IP}(\omega)\}$ for binary $\hat{x} \in X$. So we now need to relate facets of $\{y(\omega) \mid (\hat{x}, y(\omega)) \in Y_{IP}^c(\omega)\}$ with those of $Y_{IP}^c(\omega)$. We use the following result on lifting of facets (valid inequalities) that can be found in Araoz et al. (1984) and apply it to our setting:

THEOREM 4.4. *The inequality $\alpha(\omega)^\top x + \lambda\beta(\omega)^\top y(\omega) \le \lambda g(\omega, \hat{x}, \beta(\omega)) + \alpha(\omega)^\top \hat{x}$, where $\lambda \ge 0$, is a facet extension of $\beta(\omega)^\top y(\omega) \le g(\omega, \hat{x}, \beta(\omega))$ if and only if $\lambda > 0$ and $\alpha(\omega)/\lambda$ is a vertex of the polyhedron*

$$\{\alpha(\omega) \mid \alpha(\omega)^\top x + \beta(\omega)^\top y(\omega) \le g(\omega, \hat{x}, \beta(\omega)) \text{ is valid for } Y^c_{IP}(\omega)\} \tag{14}$$

of valid inequalities; or $\lambda = 0$ and $\alpha(\omega)$ is an extreme ray of (14).

Given SP2 with binary first-stage, Theorem (4.4) allows for valid inequalities of the form $\beta'(\omega)^\top y(\omega) \le g(\omega, \hat{x}, \beta(\omega))$ to be lifted to valid inequalities for $Y^c_{IP}(\omega)$ if the extreme points of (14) are known. The lift-and-project cut generation technique (e.g. Balas et al. (1993), Carøe (1998)) provides a way for computing the extreme points of (14) as follows. Let $\hat{x} \in X$ and $\hat{y}(\omega) \in Y_{LP}(\omega, \hat{x})$ be given. Suppose that $\beta(\omega)^\top y(\omega) \le g(\omega, \hat{x}, \beta(\omega))$ is a valid inequality for $Y^c_{IP}(\omega, \hat{x})$, that is, we have an FD cut. Then a lifted FD cut (FD-L cut) of the form $\alpha(\omega)^\top x + \beta(\omega)^\top y(\omega) \le g(\omega, \hat{x}, \beta(\omega)) + \alpha(\omega)^\top \hat{x}$ that is valid for $Y^c_{IP}(\omega)$ can be found by solving the following LP:

$$\begin{aligned}
\underset{\alpha(\omega)\in\Pi^\alpha}{\text{Max}}\ & \alpha(\omega)^\top \hat{x} \\
\text{s.t.}\ & \alpha(\omega) - u_{01}^\top A - u_{02}^\top T(\omega) + u_{03} e_i \ge 0 \\
& -u_{02}^\top W(\omega) \ge -\beta(\omega) \\
& \alpha(\omega) - u_{11}^\top A - u_{12}^\top T(\omega) - u_{13} e_i \ge 0 \\
& -u_{12}^\top W(\omega) \ge -\beta(\omega) \\
& \alpha(\omega)^\top \hat{x} + u_{01}^\top b + u_{02}^\top h(\omega) - u_{13}\lfloor \hat{y}_i(\omega) \rfloor \ge -g(\omega, \hat{x}, \beta(\omega)) \\
& \alpha(\omega)^\top \hat{x} + u_{11}^\top b + u_{12}^\top h(\omega) + u_{13}\lceil \hat{y}_i(\omega) \rceil \ge -g(\omega, \hat{x}, \beta(\omega)) \\
& u_{01}, u_{02}, u_{03}, u_{11}, u_{12}, u_{13}, \ge 0
\end{aligned} \tag{15}$$

In problem (15) e_i is the i-th unit vector in $\Re^{n_1}$ and $\beta(\omega)$ and $g(\omega, \hat{x}, \beta(\omega))$ are known. The variables u_{01}, u_{02}, u_{11} and u_{12} are appropriately dimensioned vectors while u_{03} and u_{13} are scalars. We assume that the "box" constraints, $0 \le x \le 1$, on binary x and $0 \le y \le 1$ on binary y, are included in $Ax \ge b$ and $W(\omega)y(\omega) \ge h(\omega) - T(\omega)x$, respectively. The objective function maximizes the distance from the point $(\hat{x}, \hat{y}(\omega))$ to the hyperplane $\alpha(\omega)^\top x + \beta(\omega)^\top y(\omega) \le g(\omega, \hat{x}, \beta(\omega))$. Notice that at $x = \hat{x}$ the FD-L cut remains $\beta(\omega)^\top y(\omega) \le g(\omega, \hat{x}, \beta(\omega))$, which is valid for $Y^c_{IP}(\omega, \hat{x})$. So the quality of the FD-L cuts cannot be readily determined.

4.2 FD Algorithm with Lifting

We can now formally state a basic FD algorithm based on FD-L cuts. The master program at iteration k of the algorithm takes the following form:

$$\begin{aligned}
\text{Min}\ & c^\top x + \eta \\
\text{s.t.}\ & Ax \ge b \\
& \sigma_t^\top x + \eta \le \nu_t,\ t = 1, \cdots, k \\
& x \in \mathbb{B}^{n_1}.
\end{aligned} \tag{16a}$$

Recall that constraints (16a) are the *optimality* cuts. For each outcome $\omega \in \Omega$, the scenario subproblem relaxation is:

$$\begin{aligned} f_c^k(\omega, x) = \text{Min } & q(\omega)^\top y \\ \text{s.t. } & W(\omega)y \geq r(\omega) - T(\omega)x \\ & \beta^t(\omega)^\top y \geq g(\alpha^t(\omega), \beta^t(\omega), \omega) + \alpha^t(\omega)^\top x^t - \alpha^t(\omega)^\top x, \; t \in \mathcal{C}_k(\omega) \\ & y \geq 0. \end{aligned} \tag{17}$$

The second set of constraints in (17) are all the FD-L cuts generated and added to the scenario problem by iteration k. An FD algorithm based on FD-L cuts can be given as follows:

Basic FD-L Algorithm:

Step 0. Initialization. Set $k \leftarrow 1$, $L^1 \leftarrow -\infty$, $U^1 \leftarrow \infty$, and let $\epsilon > 0$ be given.

Step 1. Solve LP Relaxation. Solve LP relaxation (11) using the L-shaped method to get solution $(x^k, \{y^k(\omega)\}_{\omega\in\Omega})$ and objective value z^k. Set $L^k = z^k$. If $(x^k, \{y^k(\omega)\}_{\omega\in\Omega})$ satisfy the integer restrictions, set $x^*{=}x^k$, $U^k = z^k$ and stop, x^* is optimal. Otherwise, set $k \leftarrow k+1$ and go to Step 3.

Step 2. Solve Subproblem LPs. Solve subproblem (17) for all $\omega \in \Omega$. If $(x^k, \{y^k(\omega)\}_{\omega\in\Omega})$ satisfy integer restrictions, set $U^{k+1} \leftarrow \text{Min}\{c^\top x^k + \mathbb{E}[f_c^k(x^k, \tilde{\omega})],\ U^k\}$ and if U^{k+1} is updated, set incumbent solution $x_\epsilon^* \leftarrow x^k$, and go to Step 5.

Step 3. Solve FD cut Generation Subproblems and Add Cuts. For each $\omega \in \Omega$ such that $(x^k, y^k(\omega))$ is non-integer perform the following:
(**i**) Form and solve (13) to obtain $\beta^k(\omega)$ and $g(\alpha^k(\omega), \beta^k(\omega), \omega)$.
(**ii**) Use $\beta^k(\omega)$ and $g(\alpha^k(\omega), \beta^k(\omega), \omega)$ from Step 3(i) to form problem (15) and solve to get $\alpha^k(\omega)$.
(**iii**) Form FD-L cut $\beta^k(\omega)^\top y \leq g(\alpha^k(\omega), \beta^k(\omega), \omega) + \alpha^k(\omega)^\top x^k - \alpha^k(\omega)^\top x$ and append to subproblem (17).

Step 4. Re-Solve Subproblem LPs. Re-solve subproblem (17) for all $\omega \in \Omega$ for which an FD-L cut was added. If $(x^k, \{y^k(\omega)\}_{\omega\in\Omega})$ satisfy integer restrictions, set $U^{k+1} \leftarrow \text{Min}\{c^\top x^k + \mathbb{E}[f_c^k(\omega, x^k)],\ U^k\}$. If U^{k+1} is updated, set incumbent solution $x_\epsilon^* \leftarrow x^k$. Go to Step 5.

Step 5. Update and Solve the Master Problem. Compute an *optimality cut* using the dual multipliers from the most recently solved subproblem and add to the master program (16). Solve the master program to get optimal solution x^{k+1} and optimal value z^{k+1}. Set $L^{k+1} \leftarrow \text{Max}\{z^{k+1},\ L^k\}$. If $U^{k+1} - L^{k+1} \leq \epsilon$, stop and declare x_ϵ^* ϵ-optimal. Otherwise, $k \leftarrow k+1$ and repeat from Step 2.

4.3 Lifted FD Cut Generation

Step 3 of the basic FD-L algorithm requires the solution of problem (13). We consider solving problem (13) using the decomposition method described in subsection 3.2, but now applying Proposition 4.2. Given a non-integer point $(x^k, y^k(\omega))$ at iteration k of the FD-L algorithm, $\delta^k(\omega)$ can be maximized by solving the following problem:

$$\begin{aligned} \delta^k(\omega) = \underset{\beta(\omega)\in\Pi^\beta}{\text{Max}} \quad & \theta \\ \text{s.t.} \quad & -\theta + (y^k(\omega) - y^t(\omega))^\top \beta(\omega) \geq 0, \\ & y^t(\omega) \in \text{vert}(Y^c_{IP}(\omega, x^k)). \end{aligned} \tag{18}$$

Problem (8) can be solved using a linearly defined domain for Π^β such as $\Pi^\beta = \{\beta(\omega) \mid \beta(\omega) \leq 1, \beta(\omega) \geq 0\}$. Observe that problem (18) is smaller in size than (8). Therefore, we would expect problem (18) to be relatively easier to solve.

Let t and τ denote iteration indices for the cut generation subroutine. Then given $x^k \in X$ and $y^k(\omega) \in Y_{LP}(\omega, x^k)$, at iteration τ the subroutine master program takes the following form:

$$\delta^k_\tau(\omega) = \underset{\beta(\omega)\in\Pi^\beta}{\text{Max}} \quad \theta \tag{19a}$$

$$\text{s.t.} \quad -\theta + (y^k(\omega) - y^t(\omega))^\top \beta(\omega) \geq 0, \; t = 1, \cdots, \tau. \tag{19b}$$

Given an optimal solution $(\theta^t, \beta^t(\omega))$ to (19) at iteration t, $y^t(\omega)$ is the optimal solution to the following subproblem:

$$\begin{aligned} g(\beta^t(\omega), \omega) = \text{Max} \quad & \beta^t(\omega)^\top y(\omega) \\ \text{s.t.} \quad & y(\omega) \in Y_{IP}(\omega, x^t). \end{aligned} \tag{20}$$

An FD cut generation subroutine for binary first-stage can be given as follows:

FD-L Cut Generation Subroutine:

Step 0. Initialization.
Set $t \leftarrow 1$, $\ell^1 \leftarrow -\infty$, $u^1 \leftarrow \infty$, and choose $\epsilon' > 0$ and $\beta^1(\omega) \in \Pi^\beta$.

Step 1. Solve Subproblem and Compute Lower Bound. Use $\beta^t(\omega)$ to form and solve subproblem (20) to get solution $y^t(\omega)$ and objective value $g_t(\beta^t(\omega), \omega)$. Let $d_t = (y^k(\omega) - y^t(\omega))^\top \beta^t(\omega) - g_t(\beta^t(\omega), \omega)$. Set $\ell^{t+1} \leftarrow \text{Max}\{d_t, \; \ell^t\}$. If ℓ^{t+1} is updated, set incumbent solution $(\beta^*(\omega), g^*(\beta^t(\omega), \omega)) = (\beta^t(\omega), g_t(\beta^t(\omega), \omega))$.

Step 2. Solve Master Problem. Using current non-integer solution $(x^k, y^k(\omega))$ and subproblem (20) solution $y^t(\omega)$ to form and add constraint (19b) to master program. Solve master program to get an optimal solution $(\theta^t, \beta^t(\omega))$. Set $u_{k+1} \leftarrow \text{Min}\{\theta^t, \; u^t\}$. If $u^{t+1} - \ell^{t+1} \leq \epsilon'$, stop and declare incumbent solution ϵ'-optimal. Otherwise, set $t \leftarrow t + 1$ and go to Step 1.

There are tradeoffs between the FD and FD-L algorithms. For example, the FD algorithm involves solving one larger cut generation problem (8) while the FD-L algorithm requires solving a smaller cut generation problem (18) plus an LP (15) for cut lifting. So we would expect the FD-L algorithm to gain in computational time since solving LPs is relatively easier. However, the number of iterations under FD-L may be larger since the cuts are generated based on a fixed x^k, and lifting does not provide a guarantee on the quality of the lifted cuts.

4.4 Upper Bounding

Computing an upper bound in algorithms for SP2 is generally expensive since it involves the solution of all scenario subproblems, which are integer programs. In the case of the FD-L algorithm however, we can exploit the cut generation subroutine to enable us to compute an upper bound at every iteration of the algorithm. Observe that in Step 1 of FD-L Cut Generation Subroutine we are solving for an integer feasible solution $y^t(\omega)$ to subproblem (2). Therefore, we propose computing upper bounds by selecting the best solution from $\{y^t(\omega)\}_{t=1}^{\tau}$, which the set of solutions generated in the course of computing an FD cut.

Let $w_t(\omega, x^k)$ denote the best subproblem objective value for scenario ω at iteration t of FD-L Cut Generation Subroutine. To begin, set $w^1(\omega, x^k) = \infty$ in Step 0 at iteration $t = 1$. Given the solution $y^t(\omega)$ at iteration $t > 1$ in Step 2, update $w^{t+1}(\omega, x^k) = \text{Min}\{w^t(\omega, x^k), q(\omega)^\top y^t(\omega)\}$. Let $w^*(\omega, x^k)$ and $y^*(\omega)$ be the best $w^t(\omega, x^k)$ value and corresponding $y^t(\omega)$ solution at termination of the cut generation subroutine. Also let $\Omega^k_{cut} \subseteq \Omega$ denote the subset of scenarios at iteration k of the FD-L algorithm for which an FD cut is generated. Then in Step 4 of FD-L algorithm an alternative upper bound U^k_a at iteration k can be calculated as follows:

$$U^k_a = c^\top x^k + \sum_{\omega \in \Omega \setminus \Omega^k_{cut}} p_\omega f^k_c(\omega, x^k) + \sum_{\omega \in \Omega^k_{cut}} p_\omega w^*(\omega, x^k) \tag{21}$$

If in Step 4 of FD-L algorithm all the scenario subproblem LPs (17) yield integer solutions, the upper bound U^{k+1} can be calculated as

$$U^{k+1} = \text{Min}\{c^\top x^k + \mathbb{E}[f^k_c(\omega, x^k)],\ U^k_a,\ U^k\}$$

and the incumbent updated accordingly. Otherwise, if a cut is generated for at least one scenario, then the upper bound can be updated using

$$U^{k+1} = \text{Min}\{U^k_a,\ U^k\}.$$

So unlike the FD algorithm, the FD-L algorithm offers the advantage of readily computing upper bounds at every iteration of the algorithm.

5 Computational Results

We implemented the FD and FD-L algorithms using CPLEX 11.0 Callable Library ILOG (2007) in Microsoft Visual C++ 6.0 and applied the algorithms to a variety of test instances from the literature. The aim was to gain insights into the computational performance of the algorithms based on standard test instances. We considered two types of test problems: simple standard SMIP test instances (Section 5.1) and realistic test instances from the literature (Section 5.2). We used CPLEX MIP and LP solvers to optimize the master program and subproblems in both algorithms. The instances were run to optimality or stopped when a CPU time limit of 3600 seconds (1 hour) was reached. Stopping tolerances of $\epsilon = 10^{-6}$ were used for both algorithms and $\epsilon' = 10^{-3}$ was used for the cut generation subroutines. As a benchmark, the CPLEX MIP solver was applied to the extensive form (EF2) for comparison using CPLEX default settings. All the test instances are available on the author's web site.

5.1 Simple SMIP Example Test Instances

The first set of instances (Set 1) were created based on the following variation of the well-studied example stochastic knapsack problem (SKP) instance from Schultz et al. (1998):

$$\begin{aligned} &\text{Min} - 1.5x_1 - 4x_2 + \mathbb{E}[f(x_1, x_2, \omega_1, \omega_2)] \\ &\text{s.t.} \ - x_1 \geq -1, -x_2 \geq -1 \\ &\qquad x_1, x_2 \in \mathbb{B}, \end{aligned} \tag{22}$$

where,

$$\begin{aligned} f(x_1, x_2, \omega_1, \omega_2) = \text{Min} & - 16y_1 - 19y_2 - 23y_3 - 28y_4 \\ \text{s.t.} & - 2y_1 - 3y_2 - 4y_3 - 5y_4 \geq -\omega_1 + \frac{1}{3}x_1 + \frac{2}{3}x_2 \\ & - 6y_1 - 1y_2 - 3y_3 - 2y_4 \geq -\omega_2 + \frac{2}{3}x_1 + \frac{1}{3}x_2 \\ & - y_1 \geq -1 \\ & - y_2 \geq -1 \\ & - y_3 \geq -1 \\ & - y_4 \geq -1 \\ & y_1, y_2, y_3, y_4 \in \mathbb{B}. \end{aligned}$$

This problem has binary first-stage and binary second-stage variables. The random variable (ω_1, ω_2) is uniformly distributed and varying number of scenarios (36, 121, 441, 1681 and 2601) were created by taking Ω as equidistant lattice points in $[5, 15] \times [5, 15]$. The instance characteristics are given in Table 1. The columns of the table are as follows: 'Scens' is the number of scenarios; 'Bvars' is the number binary variables; 'Constr' is number of constraints; 'Nzeros' is the number of nonzeros in the constraint matrix; z_{LP} is the LP-relaxation optimal value; z_{IP} is the optimal value; and $\%z_{Gap}$ is the LP-relaxation gap.

Table 1: Problem characteristics for Set 1

Instance	Scens	Bvars	Constr	Nzeros	z_{LP}	z_{IP}	$\%z_{Gap}$
SKP36b	36	146	74	434	-60.5434	-55.2778	9.53
SKP121b	121	486	244	1454	-61.1822	-56.2810	8.71
SKP441b	441	1766	884	5294	-61.4979	-55.2517	11.30
SKP1681b	1681	6726	3364	20174	-61.6504	-54.7139	12.68
SKP2601b	2601	10406	5204	31214	-61.6802	-54.6045	12.96

The computational experiments were conducted on a DELL Optiplex 960 with a Intel Core2 dual CPU E8600 at 3.33GHz with 8GB RAM. The benchmark results are reported in Table 2. In the table, 'CPLEX % Gap' is the percent optimality gap reported by the CPLEX MIP solver upon termination, and 'Nodes' is the number of branch-and-bound nodes. Notice that CPLEX solves the first three instances to optimality but fails to close the gap on the last two instances within the time limit.

Table 2: Benchmark results for Set 1 using CPLEX on EF2

Instance	z_{IP}	CPLEX % Gap	Nodes	CPU (secs)
SKP36b	-55.2778	0.00	6	0.22
SKP121b	-56.2810	0.00	3	0.08
SKP441b	-55.2517	0.00	0	0.25
SKP1681b	-54.7139	0.73	88236	3600.00*
SKP2601b	-54.6044	0.59	61826	3600.00*

*Time limit (1 hour) reached.

The results for FD and FD-L are reported in Table 3. In the Table, the columns 'Iters' and 'FD Cuts' give the number of algorithm iterations and number of FD cuts generated, respectively. Both FD and FD-L solve all the instances to optimality. The CPU times increase with instance size, but the FD methods perform better than CPLEX on EF2 on the larger instances, an indication that decomposition is worthwhile for the larger instances. In comparing FD to FD-L, FD-L provides better CPU times for all the instances with a gain of about 25%. This means that for these instances the *lifted* FD cuts provide better performance. However, observe that the number of iterations and cuts generated under the FD-L algorithm is larger as expected.

Table 3: Computational results for Set 1

		FD Algorithm			FD-L Algorithm		
Instance	z_{IP}	Iters	FD Cuts	CPU (secs)	Iters	FD Cuts	CPU (secs)
SKP36b	-55.278	13	114	0.61	15	134	0.47
SKP121b	-56.281	10	323	1.11	10	384	0.90
SKP441b	-55.252	12	1407	4.92	12	1574	3.51
SKP1681b	-54.713	13	5048	19.47	14	6415	14.25
SKP2601b	-54.605	12	8004	30.46	12	9954	22.00

The second set of instances (Set 2) were created by modifying the second-stage problem in Set 1 as in Ahmed et al. (2004) as follows:

$$\begin{aligned} f(x_1, x_2, \omega_1, \omega_2) = \text{Min} & -16y_1 - 19y_2 - 23y_3 - 28y_4 \\ \text{s.t.} & -2y_1 - 3y_2 - 4y_3 - 5y_4 - 1y_5 \geq -\omega_1 \\ & -6y_1 - 1y_2 - 3y_3 - 2y_4 - 1y_6 \geq -\omega_2 \\ & 1.9706y_5 - 0.9706y_6 \geq \frac{1}{3}x_1 + \frac{2}{3}x_2 \\ & -0.9706y_5 + 1.9706y_6 \geq \frac{2}{3}x_1 + \frac{1}{3}x_2 \\ & -y_1 \geq -1 \\ & -y_2 \geq -1 \\ & -y_3 \geq -1 \\ & -y_4 \geq -1 \\ & y_1, y_2, y_3, y_4 \in \mathbb{B},\ y_5, y_6 \in [0,1]. \end{aligned}$$

The instance is a slight variation of the instance in Ahmed et al. (2004), which has $y_5, y_6 \in [0, 5]$. So the second set of instances have binary first-stage but mixed-binary second-stage. The instance characteristics for Set 2 are given in Table 4. In the table, 'Cvars' is the number continuous variables. The benchmark results are reported in Table 5. As shown in the table, CPLEX solves the first three instances to optimality but fails to close the gap on the last two instances within the time limit. The last two instances in Set 2 are relatively more difficult to solve compared to the corresponding instances in Set 1. This is indicated by the larger number of branch-and-bound nodes that CPLEX explores in the two instances in Set 2.

Table 4: Problem characteristics for Set 2

Instance	Scens	Bvars	Cvars	Constr	Dens	z_{LP}	z_{IP}	$\%z_{Gap}$
SKP36m	36	146	72	74	650	-60.8730	-55.2778	10.12
SKP121m	121	486	242	244	2180	-61.5623	-56.2810	9.38
SKP441m	441	1766	882	884	7940	-61.9039	-55.2517	12.04
SKP1681m	1681	6726	3362	3364	30260	-62.0611	-54.7139	13.43
SKP2601m	2601	10406	5202	5204	46820	-62.0927	-54.8674	13.17

Table 5: Benchmark results for Set 2 using CPLEX on EF2

Instance	z_{IP}	CPLEX % Gap	Nodes	CPU (secs)
SKP36m	-55.2778	0.00	19	0.05
SKP121m	-56.2810	0.00	0	0.06
SKP441m	-55.2517	0.00	2	0.20
SKP1681m	-54.7139	0.02	270580	3600.00*
SKP2601m	-54.8674	0.12	188130	3600.00*

*Time limit (1 hour) reached.

The results for FD and FD-L on Set 2 are reported in Table 6. Both FD and FD-L solve all the instances to optimality. FD-L provides better CPU times overall (about

18% gain) compared to FD, again an indication of the effectiveness of the FD-L cuts on these instances. Overall, FD-L performs better than FD on both sets of instances. Also, FD-L was observed to provide better upper bounds in early iterations than FD.

Table 6: Computational results for Set 2

		FD Algorithm			FD-L Algorithm		
Instance	Z_{IP}	Iters	FD Cuts	CPU (secs)	Iters	FD Cuts	CPU (secs)
SKP36m	-55.278	12	99	0.49	13	127	0.59
SKP121m	-56.281	10	337	1.45	10	379	0.99
SKP441m	-55.256	11	1215	5.13	12	1564	3.58
SKP1681m	-54.714	14	5091	21.92	14	5646	15.73
SKP2601m	-54.867	13	7711	31.63	14	9498	24.97

5.2 Stochastic Server Location Problem Instances

We also tested the FD and FD-L algorithms on large-scale realistic stochastic server location problem (SSLP) instances reported in Ntaimo and Tanner (2008). The SSLP involves how to make optimal strategic decisions regarding where to locate 'servers' here-and-now in the face of future uncertainty regarding resource demand based on whether or not 'clients' will be available in the future for service. This problem arises is many applications as electric power management, internet services, and telecommunications. The SSLP instances we consider were previously solved using the disjunctive decomposition ($D2$) algorithm Sen and Higle (2005) and their characteristics are given in Table 7. This test set has a total of 50 instances: 10 problem sizes with 5 replications each. The instances are named 'SSLP$m.n.S$', where m is the number of potential server locations, n is the number of potential clients, and $S= |\Omega|$ is the number of scenarios.

Table 7: SSLP Instance Characteristics.

	EF				Second-Stage		
Instance	Cons	Bins	Cvars	Total Vars	Cons	Bins	Cvars
SSLP5.25.50	1,501	6,255	250	6,505	30	130	5
SSLP5.25.100	3,001	12,505	500	13,005	30	130	5
SSLP10.50.50	3,001	25,010	500	25,510	60	510	10
SSLP10.50.100	6,001	50,010	1,000	51,010	60	510	10
SSLP10.50.500	30,001	250,010	5,000	255,010	60	510	10
SSLP15.45.5	301	3,390	75	3,465	60	690	15
SSLP15.45.10	601	6,765	150	6,915	60	690	15
SSLP15.45.15	901	10,140	225	10,365	60	690	15
SSLP15.45.20	1201	13,515	300	13,815	60	690	15
SSLP15.45.25	1501	16,890	375	17,265	60	690	15

We performed computations on an Optiplex GX620 computer with a Pentium D processor running at 3.0Hz with 3.5GB RAM. Due to the large-scale nature of the instances, we generated added the Laporte and Louveaux [1993] (L2) optimality cut to the master program in order to close gap between the lower and upper bounds when

x^k stabilized (not changing for ten consecutive iterations). This was also done in Ntaimo and Tanner (2008) under the $D2$ algorithms. The benchmark results using CPLEX on EF2 are given in Table 8. Since this test set has five replications for same size instances, we report the minimum (Min), maximum (Max) and average (Avg) values for the number of branch-and-bound nodes and CPU time. The results show that CPLEX is unable to solve several instances to optimality within the time limit, an indication of problem difficulty. Thus decomposition methods are necessarily.

Table 8: Results for CPLEX on EF2 for SSLP.

	Nodes			CPU (secs)			CPLEX % Gap
Instance	Min	Max	Avg	Min	Max	Avg	Avg
SSLP5.25.50	0	462	125.6	8.86	38.03	23.65	0.00
SSLP5.25.100	11	2020	562.6	38.00	223.68	99.05	0.00
SSLP10.50.50	0	44491	24426.4	1055.28	3600.00*	1997.18	0.01
SSLP10.50.100	6261	24024	14582.6	1153.91	3600.00*	2670.61	0.01
SSLP10.50.500	0	488	97.6	3600.00*	3600.00*	3611.15	8.15
SSLP15.45.5	62	51613	11573.6	2.64	353.27	81.44	0.00
SSLP15.45.10	0	437451	176862.8	55.24	3600.00*	1664.81	0.01
SSLP15.45.15	12811	539143	280891	321.36	3600.00*	2662.08	0.20
SSLP15.45.20	22663	337996	111826.4	1282.12	3600.00*	2279.05	0.06
SSLP15.45.25	99076	219049	146473.8	2780.02	3600.00*	3436.04	0.19

*Time limit (1 hour) reached.

The results for using the FD and FD-L algorithms are reported in Table 9 and Table 10, respectively. Both FD and FD-L perform better than CPLEX on EF2 on all the instances, demonstrating the effectiveness of the decomposition method. The performance of FD relative to FD-L is comparable, although FD-L performs slightly better than FD on more instances. Again we observed FD-L computing better upper bounds in early iterations than FD.

Table 9: Computational Results on SSLP using FD.

	FD Algorithm			CPU (secs)		
Instance	FD Iters	FD Cuts	L2 Cuts	Min	Max	Avg
SSLP5.25.50	2.80	3.40	0.00	0.37	0.69	0.54
SSLP5.25.100	6.40	15.00	0.20	0.52	3.19	1.24
SSLP10.50.50	30.60	495.60	1.00	40.31	79.02	59.50
SSLP10.50.100	32.40	936.60	1.00	78.39	134.47	102.75
SSLP10.50.500	28.40	4669.60	1.00	376.54	517.81	451.80
SSLP15.45.5	94.60	321.20	1.00	7.99	526.85	136.67
SSLP15.45.10	77.75	588.00	1.00	17.58	376.75	171.45
SSLP15.45.15	118.20	1318.40	1.00	22.61	771.80	338.37
SSLP15.45.20	78.40	1139.20	1.00	100.06	300.25	166.72
SSLP15.45.25	83.50	1393.00	1.00	85.41	244.59	168.69

Finally, we compared the FD algorithms to the D2 algorithm applied to the SSLP instances. Recall that the two classes of SMIP algorithms are totally different. The goal was to compare FD and FD-L to another decomposition method. The results are

Table 10: Computational Results on SSLP using FD-L.

	FD-L Algorithm			CPU (secs)		
Instance	FD-L Iters	FD-L Cuts	L2 Cuts	Min	Max	Avg
SSLP5.25.50	2.40	2.80	0.00	0.36	0.70	0.52
SSLP5.25.100	6.00	12.80	0.00	0.37	2.56	1.10
SSLP10.50.50	31.40	503.20	1.00	37.51	75.79	57.74
SSLP10.50.100	31.20	899.40	1.00	69.44	108.48	91.57
SSLP10.50.500	28.60	4601.60	1.00	345.20	476.56	416.07
SSLP15.45.5	94.80	327.20	1.00	7.83	553.60	138.61
SSLP15.45.10	84.25	637.50	1.00	27.39	404.66	181.53
SSLP15.45.15	117.00	1306.60	1.00	21.58	723.06	333.20
SSLP15.45.20	80.60	1164.00	1.00	102.71	247.63	166.69
SSLP15.45.25	83.25	1383.00	1.00	82.57	201.81	151.61

reported in Table 11. Compared to FD and FD-L, it is interesting to observe that the D2 algorithm performs better on the first five problems. However, FD and FD-L perform better on the last five problems, which have larger first-stage decision dimension space. Also, FD-L computes better upper bounds in early iterations than $D2$. In generally, the FD method outperforms the direct method (CPLEX on EF2) and the ($D2$) algorithm on large-scale instances.

Table 11: Computational Results on SSLP using $D2$.

	$D2$ Algorithm			CPU (secs)		
Instance	D2 Iters	D2 Cuts	L2 Cuts	Min	Max	Avg
SSLP5.25.50	18.33	33.33	0.67	0.08	0.22	0.17
SSLP5.25.100	18.33	66.67	0.67	0.16	0.33	0.25
SSLP10.50.50	249.67	233.33	1.00	17.17	47.44	27.75
SSLP10.50.100	251.00	616.67	1.00	18.51	96.21	37.88
SSLP10.50.500	268.33	2416.67	1.00	61.58	96.21	80.84
SSLP15.45.5	281.33	248.33	3.33	9.67	1555.45	306.53
SSLP15.45.10	608.17	365.00	2.00	38.13	1400.89	581.91
SSLP15.45.15	426.67	835.00	2.67	24.35	2006.32	552.25
SSLP15.45.20	312.40	668.00	1.80	49.01	1613.02	454.62
SSLP15.45.25	375.60	960.00	3.00	229.46	615.95	433.17

6 Conclusion

This paper introduces a new cutting plane method for two-stage stochastic mixed-integer programming (SMIP) called Fenchel decomposition (FD). FD is based on a class of valid inequalities termed, FD cuts, which are derived based on Fenchel cutting planes from integer programming. We derive FD cuts based on both the first-stage and second-stage variables and devise an FD algorithm for SMIP and establish finite convergence for binary first-stage. As an alternative, we also derive FD cuts based on the second-stage variables only, and borrow an idea from disjunctive programming to lift the cuts to the higher dimension space including the first-stage variables. We then devise the

FD-L algorithm based on the lifted FD cuts. Preliminary computational results are promising and show the lifted FD cuts to have better performance than the regular FD cuts in general. Furthermore, both the FD and FD-L algorithms outperform a standard direct solver and the disjunctive decomposition method on some large-scale instances. In terms of future work, this paper opens up several directions of research. For example, specialized FD cut generation subroutines need to be devised and implemented for specific applications to accelerate FD cut generation. Since cutting planes are generally more effective when implemented within a branch-and-bound scheme, the FD cuts can be incorporated in branch-and-cut algorithms for SMIP. Finally, parallel/distributed computer implementations of the FD algorithms can be explored.

Acknowledgments. This research was supported in part by a grant (No. 0720470) from the National Science Foundation.

References

Ahmed, S., Tawarmalani, M. and Sahinidis, N. V.: 2004, A finite branch and bound algorithm for two-stage stochastic integer programs, *Mathematical Programming* **100**, 355–377.

Araoz, J., Edmonds, J. and Griffin, V.: 1984, Lifting the facets of polyhedra, *in* W. R. Pulleyblank (ed.), *Progress in Combinatorial Optimization*, Academic Press, New York, pp. 3–12.

Balas, E.: 1975, Disjunctive programming: cutting planes from logical conditions, *in* O. Mangasarian, R. Meyer and S. Robinson (eds), *Nonlinear Programming 2*, Academic Press, New York.

Balas, E., Ceria, E. S. and Cornuéjols, G.: 1993, A lift-and-project cutting plane algorithm for mixed 0-1 integer programs, *Mathematical Programming* **58**, 295–324.

Benders, J. F.: 1962, Partitioning procedures for solving mixed-variable programming problems, *Numerische Mathematik* **4**, 238–252.

Birge, J. R. and Louveaux, F. V.: 1997, *Introduction to Stochastic Programming*, Springer, New York.

Blair, C. and Jeroslow, R.: 1978, A converse for disjunctive constraints, *Journal of Optimization Theory and Applications* **25**, 195–206.

Blair, C. and Jeroslow, R.: 1982, The value function of an integer program, *Mathematical Programming* **23**, 237–273.

Boyd, E. A.: 1993, Generating Fenchel cutting planes for knapsack polyhedra, *SIAM Journal on Optimization* **3(4)**, 734–750.

Boyd, E. A.: 1994a, Fenchel cuts for integer programs, *Operations Research* **42(1)**, 53–64.

Boyd, E. A.: 1994b, Solving 0/1 integer programs with enumeration cutting planes, *Annals of Operations Research* **50**, 61–72.

Boyd, E. A.: 1995, On the convergence of Fenchel cutting planes in mixed-integer programming, *SIAM Journal on Optimization* **5(2)**, 421–435.

Carøe, C. C.: 1998, *Decomposition in Stochastic Integer Programming*, Ph.D. thesis, Dept. of Operations Research, University of Copenhagen, Denmark.

Carøe, C. C. and Tind, J.: 1997, A cutting-plane approach to mixed 0-1 stochastic integer programs, *European Journal of Operational Research* **101**, 306–316.

ILOG: 2007, *CPLEX 11.0 ILOG CPLEX Callable Library 11.0 Reference Manual*, ILOG, France.

Klein Haneveld, W. K. and van der Vlerk, M. H.: 1999, Stochastic integer programming: general models and algorithms, *Annals of Operations Research* **85**, 39–57.

Louveaux, F. V. and Schultz, R.: 2003, Stochastic integer programming, *in* A. Ruszczyn'ski and A. Shapiro (eds), *Stochastic Programming (Handbooks in Operations Research and Management Science 10)*, Elsevier, chapter 6, pp. 213–266.

Ntaimo, L. and Sen, S.: 2008, A comparative study of decomposition algorithms for stochastic combinatorial optimization, *Computational Optimization and Applications Journal* **40(3)**, 299–319.

Ntaimo, L. and Tanner, M. W.: 2008, Computations with disjunctive cuts for two-stage stochastic mixed 0-1 integer programs, *Journal of Global Optimization,* **41(3)**, 365–384.

Ramos, M. T. and Sáez, J.: 2005, Solving capacitated facility location problems by fenchel cutting planes, *Journal of the Operations Research Society* **56**, 297–306.

Rockafellar, R. T.: 1970, *Convex Analysis*, Princeton University Press, Princeton, New Jersey.

Ruszczyn'ski, A. and Shapiro, A. (eds): 2003, *Stochastic Programming*, Vol. 10 of *Handbooks in Operations Research and Management Science*, Elsevier.

Sáez, J.: 2000, Solving linear programming relaxations associated with Lagrangean relaxations by Fenchel cutting planes, *European Journal of Operational Research* **12**, 609–626.

Schultz, R.: 1993, Continuity properties of expectation functions in stochastic integer programming, *Mathematics of Operations Research* **18**, 578–589.

Schultz, R.: 2003, Stochastic programming with integer variables, *Mathematical Programming* **97**, 285–309.

Schultz, R., Stougie, L. and van der Vlerk, M. H.: 1996, Two-stage stochastic integer programming: A survey, *Statistica Neerlandica* **50**, 404–416.

Schultz, R., Stougie, L. and van der Vlerk, M. H.: 1998, Solving stochastic programs with integer recourse by enumeration: a framework using Gröbner basis reduction, *Mathematical Programming* **83(2)**, 71–94.

Sen, S.: 2005, Algorithms for stochastic mixed-integer programming models, *in* K. Aardal, G. Nemhauser and R. Weismantel (eds), *Handbooks in Operations Research and Management Science, Volume 12: Discrete Optimization*, Elsevier, Dordrecht, The Netherlands, chapter 9.

Sen, S. and Higle, J. L.: 2005, The C3 theorem and a D2 algorithm for large scale stochastic mixed-integer programming: Set convexification, *Mathematical Programming* **104(1)**, 1–20.

Sen, S., Higle, J. L. and Ntaimo, L.: 2002, A summary and illustration of disjunctive decomposition with set convexification, *in* D. L. Woodruff (ed.), *Stochastic Integer Programming and Network Interdiction Models*, Kluwer Academic Press, Dordrecht, The Netherlands, chapter 6, pp. 105–123.

Sen, S. and Sherali, H. D.: 2006, Decomposition with branch-and-cut approaches for two stage stochastic mixed-integer programming, *Mathematical Programming* **106(2)**, 203–223.

Shapiro, A., Dentcheva, D. and Ruszczyn'ski, A. (eds): 2009, *Lectures on Stochastic Programming*, SIAM, Philadelphia. http://www2.isye.gatech.edu/people/faculty/Alex_Shapiro/SPbook.pdf.

Sherali, H. D. and Shetty, C. M.: 1980, Optimization with disjunctive constraints, *Lecture Notes in Economics and Math. Systems,* **181**, 411–430.

Sherali, H. D. and Zhu, X.: 2006, On solving discrete two-stage stochastic programs having mixed-integer first- and second-stage variables, *Mathematical Programming* **108(2-3)**, 597–616.

Slyke, R. V. and Wets, R.-B.: 1969, L-shaped linear programs with application to optimal control and stochastic programming, *SIAM Journal on Applied Mathematics* **17**, 638–663.

van der Vlerk, M. H.: 2007, Stochastic integer programming bibliography. http://mally.eco.rug.nl/biblio/sip.html, 1996-2007.

Wollmer, R. M.: 1980, Two stage linear programming under uncertainty with 0-1 first stage variables, *Mathematical Programming* **19**, 279–288.